Problem 1. The following psuedo-code is an implementation of the idea: (E is adjacency list)

```
1:  procedure ToplogicalSort(V, E)
2:      S := ∅
3:      Q := ∅
4:      r := array(0, 0, ··· 0) (length |V|)
5:      for e = (v_i, v_j) ∈ E do
6:          r[j] = r[j] + 1
7:      end for
8:      for i from 1 to |V| do
9:          if r[i] ≠ 0 then
10:             Q.enqueue(v_i)
11:         end if
12:     end for
13:     while Q ≠ ∅ do
14:         v = Q.dequeue(); S.append(v)
15:         V = V \ {v}
16:         for v_i where (v, v_i) ∈ E do
17:             r[i] = r[i] − 1; E \ {(v, v_i)}
18:             if r[i] = 0 then
19:                 Q.enqueue(v_i)
20:             end if
21:         end for
22:     end while
23:     return S
24: end procedure
```

The time complexity of this algorithm is $O(E)$ (5 to 7) + $O(V)$ (8 to 12) + $O(V+E)$ (13 to 22) = $O(V+E)$. Also, if G has a cycle, $Q=\emptyset$ before $V=\emptyset$. (i.e there are not possible to choose in-degree 0 vertex)

Problem 2. For convinence, view L_1 and L_2 as a function: $L_1 : X_1 \to Y_1$, $L_2 : X_2 \to Y_2$. Also, let $R : X_1 \to X_2$ is a reduction.

(a) True. Let $D(x)$ is a polynominal algorithm that solves $L_2(x)$. Then, $D(R(x))$ is a solution of $L_1(x)$ for every $x \in X_1$.

(b) Open problem. There are no public known facts that whether there exist a problem L such at $L \notin P \wedge L \notin NP\text{–}complete$.

(c) False. If $L_2 \notin NP$, $L_2 \notin NP - complete$.

(d) False. If SAT $\in P$, $NP = P$ so $co\text{–}NP = NP = P$. Refer to the problem 3.

(e) False. Suppose a problem $L : X \to \{T, F\}$, which $L(x) = F$ for every $x \in X$. Obviously, $L \in P \subseteq NP$, but that doesn't give anything meaningful about NP.

(f) False. If an $NP - complete$ problem can be solved in linear time, $NP - complete = NP = P$, but all P problems cannot be solved in linear time.

Problem 3. The proof is following:
(1) Assume that $P = NP$, and choose an arbitrary problem $L \in co\text{–}NP$.
(2) Then, $\bar{L} \in NP$, so there exists a polynominal algorithm A that solves $\bar{L}$.
(3) Since $\lambda x.\neg A(x)$ is a polynominal algorithm that solves L, $L \in P$.
(4) By (1) and (3), $co\text{–}NP \subset P$. Since it is obvious that $P \subset co\text{–}NP$, $co\text{–}NP = P$.
(5) By (1) and (4), $co\text{–}NP = NP$. Taking contrapositive gives $NP \neq co\text{–}NP \to P \neq NP$.

Problem 4. View L as a language that defined in X.
(1) Let $L <_p \bar{L}$, and R is a poly reduction function.
(2) Then, $\forall x \in X, x \in L \leftrightarrow R(x) \in \bar{L}$.
(3) $\forall x \in X, x \in \bar{L} \leftrightarrow x \notin L \leftrightarrow R(x) \notin \bar{L} \leftrightarrow R(x) \in L$
(4) Therefore, R is a poly reduction function from $\bar{L}$ to L, $\bar{L} <_p L$.
(5) The converse can be easily proven by subsituting $L = \bar{L}'$ and $\bar{L} = L'$.